TIETOVARASTOT

Tietovarastot ohjelmistokehityksessä

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto
Opinnäytetyö
2.5.2011
Juho Peltonen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

PELTONEN, JUHO: Tietovarastot ohjelmistokehityksessä

Ohjelmistotekniikan opinnäytetyö, 36 sivua

Kevät 2011

TIIVISTELMÄ

Tietovarasto sisältää yrityksen toiminnalle tärkeää tietoa. Tietovarasto säilyttää vanhat tiedot eikä niiden päälle talleteta muuta tietoa. Tietovaraston etuna on helppo ja nopea tiedonhaku.

Tietovaraston käyttäjät käyttävät tietovaraston tietoja päätösten tekoon. Monet talousmaailman päätöksistä tehdään tietokannan tietojen perusteella.

Tietovaraston tieto tulee toisista tietokannoista, joita sanotaan operatiivisiksi kannoiksi. Koska operatiivisia kantoja on useita ja niiden rakenne on erilainen, on haasteena tiedon yhtenäisyys. Tiedon muuntamista operatiivisten kantojen erilaisista rakenteista yhteen tietovaraston rakenteeseen kutsutaan integroinniksi tai ETL:ksi.

Tietokannat ovat yleensä relaatiotietokantoja, joihin tallennetaan tieto relaatiamallin mukaan. Yleisin rajapinta relaatiotietokantoihin on SQL-kieli.

Visual Studio Tools for Office (VSTO) on Microsoftin kehittämä kehitystyökalu Office-pohjaisten sovellusten tekemiseen. VSTO työkalulla voi tehdä ohjelmia esimekiksi Word ja Excel ohjelmistojen päälle.

Tässä opinnäytetyössä tehdään budjetin ennustamiseen käytettävä sovellus. Sovellus on Excel pohjainen ja se on tehty VSTO työkalulla.

Avainsanat: tietovarasto, tietokanta, operatiivinen kanta, ETL, SQL

Lahti University of Applied Sciences
Degree Programme in Information Technology

PELTONEN, JUHO: Data warehouses in software engineering

Bachelor's Thesis in software engineering 36 pages

Spring 2011

ABSTRACT

Data warehouses contain vital information for large businesses. The information in data warehouses is used by companies to make financial decisions and strategic business decisions. Data warehouses store the old existing data without overwriting it. The advantage of data warehouse is efficient and easy data queries from the database.

The data of a data warehouse comes from other databases that are called operating databases. Since there are many operating databases per one data warehouse and the structure of data in operating databases varies, the integration of data is required. The process of integrating data is called Extract, Transform, Load or ETL for short. The integration phase is usually quite complex and requires a lot of effort to make it work.

The most common type of database is a relational database. The data structure of relational database follows the relational data model. The most common interface to use relational databases is SQL language.

Visual Studio Tools for Office (VSTO) is a software development tool created by Microsoft for developing Office-based applications. With VSTO, for example, Word- and Excel-based applications can be created.

The objective of the case part of this thesis was to create an application to manage budgets. It is an Excel-based application and it was created using the VSTO tool.

Key words: data warehouse, database, operating database, ETL, SQL

SISÄLLYS

1 JOHDANTO 1

2 TIETOVARASTO 2

2.1 Toimintaympäristö 2

2.2 Tietovarasto 3

2.2.1 Historia 3

2.2.2 Relaatiotietokannat 4

2.2.3 SQL 9

2.2.4 Tietovarastoympäristö 11

2.2.5 Tietovaraston rakentaminen 12

2.2.6 Tietovaraston tekniikka 15

2.2.7 Tiedon esittäminen 16

2.2.8 Faktat ja dimensiot 18

2.3 Visual Studio Tools for Office 20

2.3.1 Vertailu muihin työkaluihin 21

2.3.2 Käyttäminen 22

3 ENNUSTESOVELLUS 24

3.1 Olemassaolevan tietokannan käyttäminen 25

3.1.1 Tietokantayhteys 26

3.2 Toimintalogiikka 28

3.3 Ulkoasu 32

4 YHTEENVETO 35

LÄHTEET 36

LIITTEET 37

1 JOHDANTO

Tietovarastot ovat tärkeitä tiedonlähteitä yrityksien päättäjille. Tietovarasto on tietokanta, joka säilöö tiedot tietynlaiseen arkkitehtuuriin. Tietovarastojen perimmäinen idea on tallettaa tietoja pysyvästi. Tietovaraston tiedot kopioidaan yleensä useasta muusta tietokannasta.

Tämän opinnäytetyön aiheena on budjetin ennustussovellus. Työ tehdään Logicalle, joka myy työn eteenpäin asiakkaalleen.

Logica Suomi on osa Logicaa. Logica on 36 maassa toimiva IT-alan yritys. Logica Suomi työllistää noin 3100 henkilöä Suomessa.

Käytännön projektina on toteuttaa budjetin ennustus sovellus Logica Suomelle. Sovellus rakennetaan Microsoft Excel -ohjelmiston päälle käyttäen Visual Studio Tools for Office työkalua. Työkalu antaa mahdollisuuden toteuttaa Excel -ohjelmia Microsoftin .NET-alustalla. Työ tehdään Logica Suomen Lahden toimipisteessä.

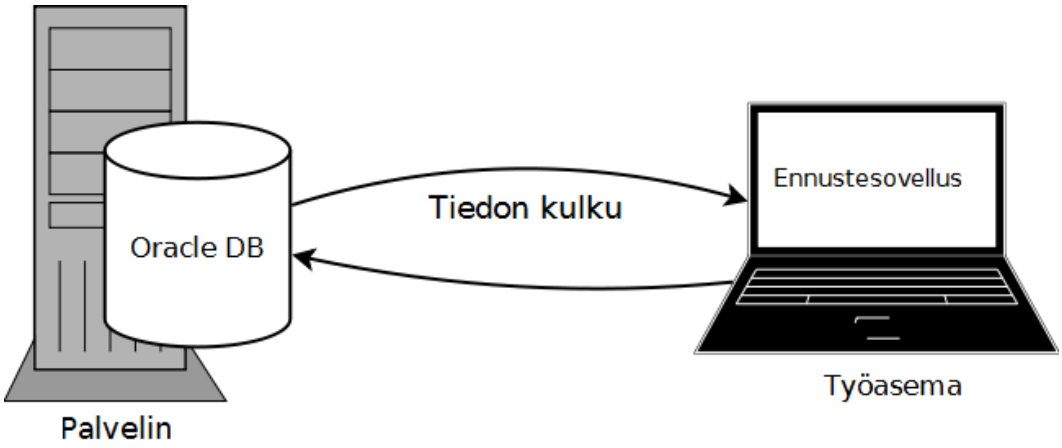
Tietovarastojen hyödyntäminen todennäköisesti kasvaa tulevaisuudessa. Työkalujen tietokantamallien kehittymisen myötä tietovarastojen suunnittelu, toteuttaminen ja ylläpito tulevat helpommaksi ja tärkeämmäksi, jolloin entistä useampi yritys pääsee hyödyntämään tietokantaa.

Tämän opinnäytetyön toinen luku käsittelee tietovarastointia sekä Microsoftin VSTO-työkalua. Kolmas luku käy läpi ennustesovelluksen toteutuksessa vastaan tulleita käytännön ongelmia. Neljäs luku sisältää yhteenvedon.

2 TIETOVARASTO

2.1 Toimintaympäristö

Tämän opinnäytetyön ympäristönä toimii kehitysympäristö, sillä sitä ei ole tarkoitus käyttää suoraan tuotantotarkoitukseen. Kehitysympäristönä toimii Windows Server 2003 -käyttöjärjestelmällä varustettu palvelin. Palvelimen suorituskyky ei aiheuta rajoitteita työlle. Kuvio 1 esittää palvelimen ja sovelluksen välisen toiminnan. Kuvion nuolet osoittava tiedon kulkusuunnan, eli tieto liikkuu sekä tietokannasta ulos, että sisään.



KUVIO 1. Ennustesovelluksen toimintaympäristö

Palvelimelle on asennettu Oracle Database -tietokantaohjelmiston versio 10.2, jolla kehitykseen käytettävä tietokanta sijaitsee. Ohjelmiston kehittäminen tapahtuu työkoneelta, jolloin tietokantayhteys otetaan verkon yli palvelimelle.

Oracle Database on Oracle Corporationin kehittämä relaatiotietokanta. Oraclen tietokannan etuna on tietokannan tehokas toiminta, hyvä ylläpidettävyys ja hyvä tuki.

Itse sovelluksen on toimittava Windows -koneella, johon on asennettu Microsoft Office -ohjelmisto.

2.2 Tietovarasto

Tieto on eräs yrityksen tärkeimmistä voimavaroista. Tietoa pidetään yllä kahdessa muodossa, operatiivisissa tietokannoissa ja tietovarastossa. Operatiiviseen tietokantaan tieto tallennetaan päivittäisessä käytössä kirjaten kaikki tapahtumat. Operatiivisen kannan tieto siirretään tietovarastoon, josta tiedon voi helposti lukea. Oikeastaan operatiivisten tietokantojen käyttäjät lisäävät kantaansa tietoa vähän kerrallaan tapahtumien tapahtuessa. Tietovaraston käyttäjät taas seuraavat tapahtumien kulkua ja tekevät korkeamman tason päätöksiä sen mukaan. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 2.)

2.2.1 Historia

Päätöksenauttamisjärjestelmiä (Decision Support System) on käytetty lähes niin kauan, kun on ollut olemassa tietokoneita. Nämä järjestelmät kehitettiin monimutkaisen tietoteknisen kehityksen tuloksena. (Inmon, Building The Data Warehouse 3rd Ed 2003, 2.)

1960-luvulla järjestelmät koostuivat lähinnä tiedostoista, joita eri ohjelmat käyttivät. Nämä ohjelmat olivat yleensä kehitetty COBOL-kielellä. Näihin aikoihin reikäkortit olivat yleisiä. Suuret datamäärät talletettiin nauha-asemiin. Nauhojen lukeminen oli hidasta, ja tieto piti lukea niistä järjestyksessä, mikä käytännössä tarkoitti, että koko nauha oli luettava joka kerta kun tietoa tarvittiin. Nauhan lukeminen saattoi kestää jopa puoli tuntia. Näiden sovellusten tekeminen oli myös huomattavan hankalaa ja järjestelmän ylläpitäminen vaikeaa. (Inmon, Building The Data Warehouse 3rd Ed 2003, 2.)

1970-luvulla kiitolevyt yleistyivät tietojärjestelmissä. Kiintolevyistä tietoa haettaessa pystyi hakemaan vain tarvitsemansa data, eikä koko levyä ollut tarvetta käydä lävitse. Lisäksi kiintolevyjen lukunopeus oli huomattavasti nauhoja nopeampaa. Tiedon hakemiseen menevä aika pieneni minuuteista millisekunteihin. (Inmon, Building The Data Warehouse 3rd Ed 2003, 2.)

Kiintolevyjen yleistyessä markkinoille alkoi ilmestyä tietokannan hallintajärjestelmiä (Database Managment System, DBMS). Nämä järjestelmät hoitivat tiedon tallentamisen ja hakemisen tehokkaasti. Tietokantajärjestelmät myös helpottivat huomattavasti päätöksentekojärjestelmien kehittämistä ohjelmoijan näkökulmasta, koska oleellinen osa ohjelman toiminnallisuutta oli jo valmiina. (Inmon, Building The Data Warehouse 3rd Ed 2003, 4.)

1980-luvun alkaessa tietokoneet olivat yleistyneet. Käyttäjät pystyivät hakemaan tietoa tietokantajärjestelmistä erilaisilla tietokantakyselykielillä. Näin tietokannan tietoa pystyi käyttämään helpommin ja monipuolisemmin. Tämän seurauksena tietokantoja ruvettiin enenevissä määrin käyttämään päätöksentekoon. (Inmon, Building The Data Warehouse 3rd Ed 2003, 5.)

Suurissa yrityksissä tietomäärien jalostaminen alkoi yleistyä. Tietoa yleensä haettiin suuremmasta tietokannasta tietyllä hakukriteerillä. Haettu tieto sitten talletettiin toiseen tietokantaa. Tieto kerättiin useasta pienemmästä tietokannasta yhteen keskustietokantaan. Tällaista ratkaisua voi jo kutsua tietovarastoksi. (Inmon, Building The Data Warehouse 3rd Ed 2003, 6.)

2.2.2 Relaatiotietokannat

Relaatiotietokanta on tietokanta, joka tallentaa tiedon tietokantatauluihin. Tietokantataulut rakentuvat riveistä ja sarakkeista. Yksi tietokantataulu sisältää normaalista vain yhdenlaista tietoa. Yksittäisen tiedon voi jakaa osiin. Nämä osat kuvataan tietokantataulun sarakkeina. (Wikipedia 2011.)

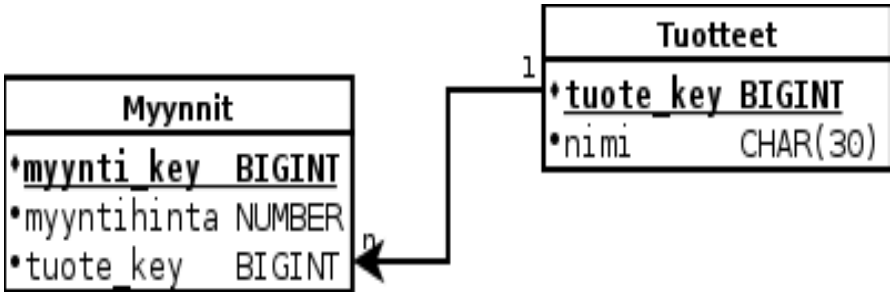
Sarake voi sisältää vain ennalta määritellyn kaltaista tietoa, kuten vaikkapa numeroita tai tekstiä. Tietokantataulussa rivi tarkoittaa yksittäistä tietoa. Näitä tietoja voi olla taulussa rajaton määrä. Sarakkeita on aina ennalta määritelty määrä. Kuviossa 2 on esimerkki

tietokantataulusta, jonka sarakkeet ovat nimi, email ja yritys. Tietokantataulu on oikeastaan kaksiulotteinen taulukko, jonka ulottuvuudet ovat sarakkeet ja rivit.(Wikipedia 2011.)

```
Table Asiakkaat:
+-------+----------------+------------------+
| nimi  | email          |                  |
+-------+----------------+------------------|
| Matti | masa@posti.fi  | Yritys Oy        |
| Maija | maija@posti.fi | Yritys Oy        |
| Janis | jan@email.com  | Corporation inc  |
+-------+----------------+------------------+
```

KUVIO 2. Esimerkki tietokantataulusta

Relaatiotietokannassa taulujen välillä on yhteyksiä eli relaatioita. Yhteydet ovat määriteltyä tietynlaisiksi haluttujen taulujen välille. Kaaviokuvan kahden taulun välisestä yhteydestä näkee kuviosta 3. Yhteydet määritellään normaalisti samaan aikaan kun taulut luodaan, eli tietokantaa tehdessä. (Wikipedia 2011.)



KUVIO 3. Kaaviokuva kahden taulun yhteydestä

Relaatiotietokannan etuna on tiedon helppo saatavuus. Tälle on edellytyksenä se, että tietokanta on suunniteltu huolellisesti. Huonosti tehtyä tietokantaa voi olla hyvinkin hankala käyttää. Hyvänä puolena on myös se, että tieto ei vahingossa katoa mihinkään, eikä korruptoidu eli muutu toiseksi ilman, että sitä tarkoituksellisesti halutaan. (Wikipedia 2011.)

Relaatiot määritellään relaatiotietokannoissa avainarvojen avulla. Tauluille määritellään yksi tai useampi kenttä pääavaimeksi (englanniksi primary key). Samaa pääavainta ei voi olla taulussa kuin kerran. Jos avaimena on useampi sarake, ei näiden sarakkeiden yhdistelmiä voi esiintyä kuin kerran (Wikipedia 2011). Kuvion 2 tauluissa myynti_key on Myynnit-taulun pääavain ja tuote_key on Tuotteet-taulun pääavain.

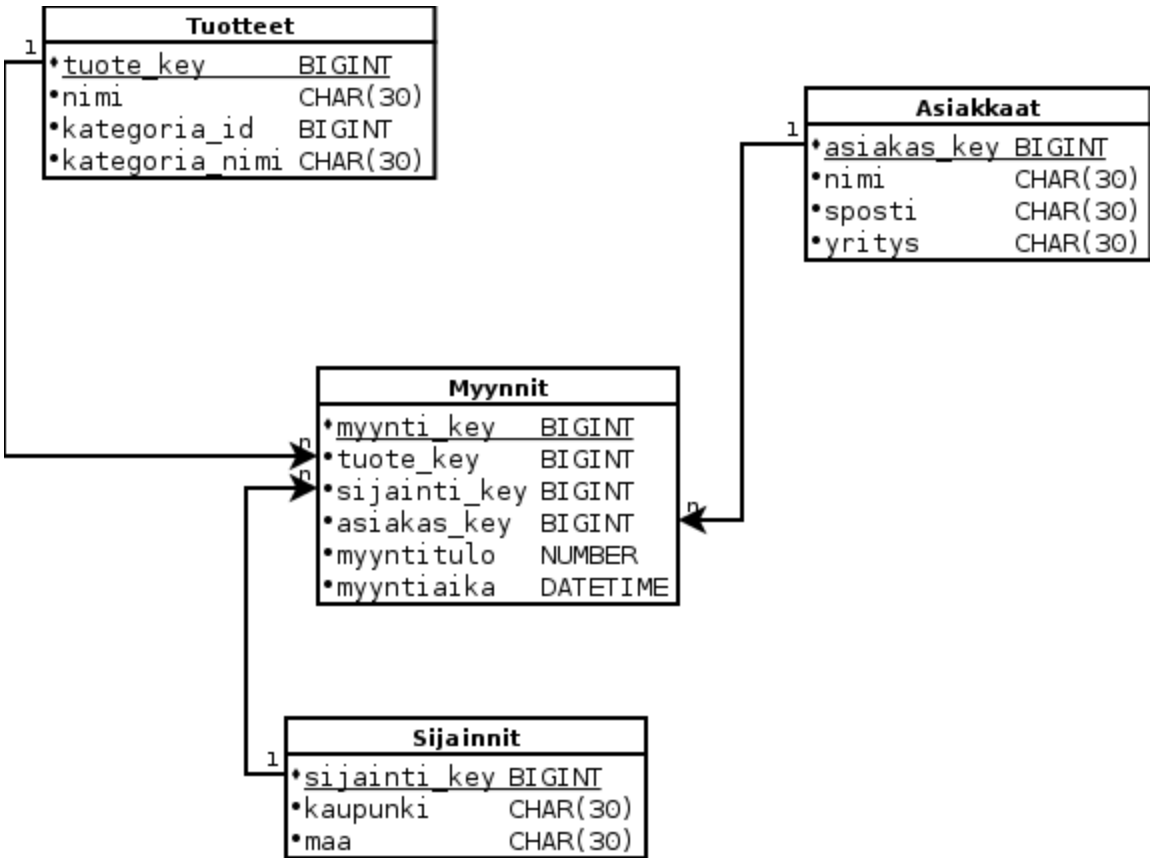
Toisesta taulusta voi viitata taulun pääavaimeen viiteavaimen (englanniksi foreign key) avulla. Viiteavain on määritelty yhdeksi taulun sarakkeeksi. Sarakkeen tietotyypin on oltava sama kuin pääavaimen omaavan taulun pääavainsarake. Jos pääavaimena on useampi sarake, on luonnollisesti viiteavaimena oltava myös useampi sarake. Kun käytössä on viiteavain, ei tauluun voi lisätä riviä, jolle ei ole vastaavaa pääavainta (Wikipedia 2011). Kuviossa 2 Myynnit-taulun tuote_key sarake on viiteavain, joka viittaa Tuotteet-taulun pääavaimeen.

Kun yhden taulun viiteavain sisältää saman arvon kun siihen liitetyn taulun sisältämän rivin pääavain, on näiden kahden rivin tieto yhdistetty. (Wikipedia 2011.)

Relaatiotietokannan rakenteelle on kehitetty useampiakin malleja. Näistä malleista yksi on tähtimalli ja toinen lumihiutalemalli. (Oracle 2002.)

Tähtimalli on melko yksinkertainen rakenne tietokannalle. Tähtimallia voi ajatella mallina, jossa keskellä on yksi taulu, jota muut taulut ympäröivät. Kaikki ympäröivät taulut ovat sidottu keskellä olevaan tauluun, mutta eivät toisiinsa. Keskellä oleva taulu sisältää varsinaisten tietoa sisältävien sarakkeiden lisäksi viiteavaimen jokaista ympärillä olevaa taulua kohden, jotta relaatio onnistuu. (Oracle 2002.)

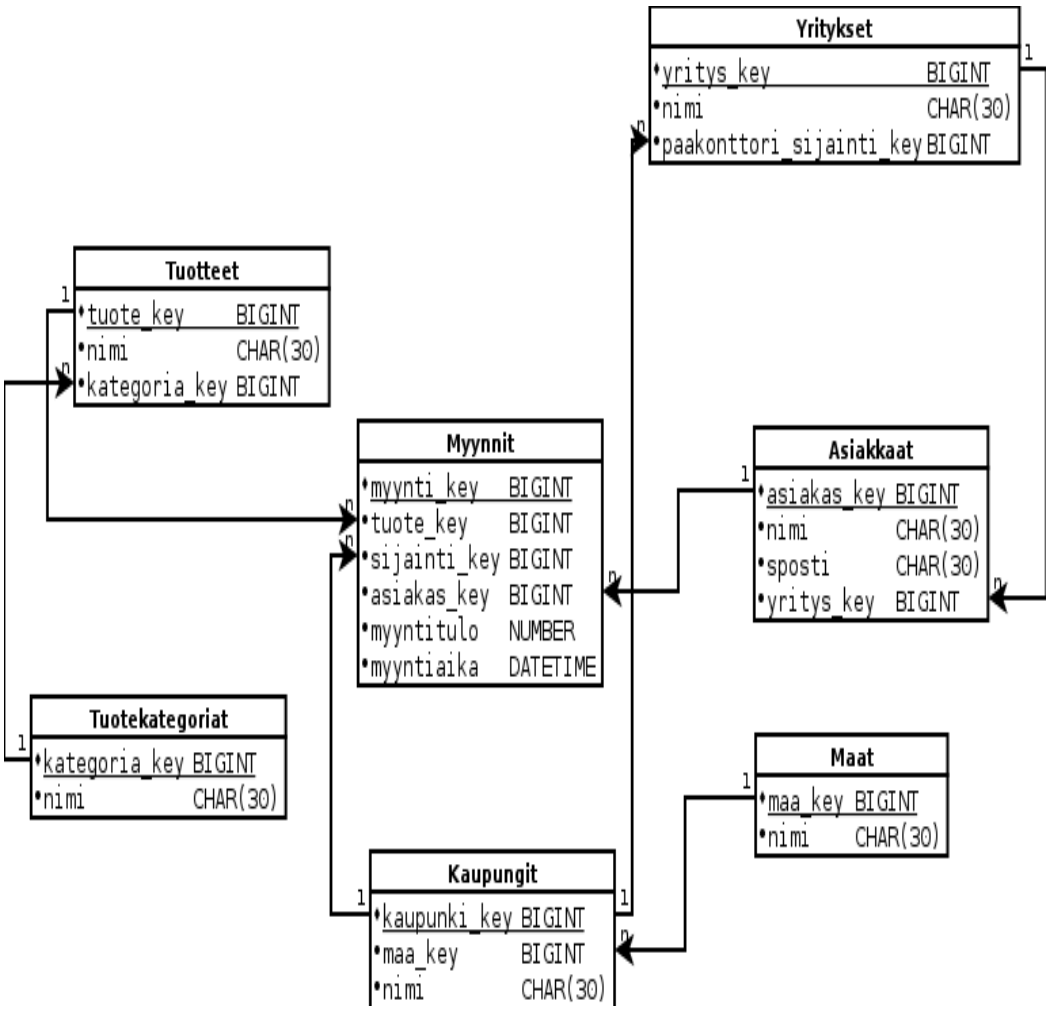
Tähtimallin yksinkertaisuus on sekä etu että haitta. Yksinkertainen rakenne tarkoitta, että tietoa on helppo ja nopea hakea, kun relaatioita on mahdollisimman vähän. Haittana on, että samaa tietoa voi helposti löytyä useampaan kertaan. Esimerkki tähtimallin kaaviokuvasta on esitelty kuviossa 4.



KUVIO 4. Esimerkki tähtimallista

Lumihiutalemalli on tähtimallia hieman monimutkaisempi. Tähtimallissa keskimmäiseen tauluun yhdistetyt taulut eivät liity toisiinsa, mutta lumihiutalemallissa voi näitä liitoksia tehdä. (Oracle 2002.)

Lumihiutalemallissa tiedon hakeminen on hieman hankalampaa kuin tähtimallista, sillä yhteyksistä voi tulla monikerroksisia. Lumihiutalemallista tiedon hakeminen voi olla myös hitaampaa, kun tietokantaohjelma joutuu yhdistelemään useamman taulun tietoja. Suurin etu lumihiutalemallissa on, että oikein tehtynä sama tieto ei toistu useassa eri paikassa. Esimerkin lumihiutalemallilla tehdystä tietokantakaaviosta näet kuviossa 5.



KUVIO 5. Esimerkki lumihiutalemallista

Tähtimallia kannattaa käyttää silloin, kun tietoa on tarkoitus lukea hyvin usein. Tähtimallin perusteella tehdyn kannan tiedon lukeminen on lumihiutalemallilla rakennettua kantaa nopeampaa.

Toisaalta, jos tietoa joutuu muokkaamaan usein, tai sitä joutuu usein lisäämään tähtimällin keskellä olevaa taulua ympäröiviin tauluihin, on lumihiutalemalli parempi. Kun lumihiutalemallissa samaa tietoa ei löydy kuin yhdestä paikasta, ei tietoa muutettaessa sitä tarvitse muuttaa kuin yhdestä paikasta. Samaten tietoa lisätessä tarvitsee sitä lisätä vain yhteen paikkaan. (Oracle 2002.)

Oikein toteutettuna lumihiutalemalliin perustuvan tietokannan tieto on normalisoitua. Tähtimallin tieto ei yleensä ole normalisoitua. Normalisoidussa tiedossa ei sama tieto toistu

useampaan kertaan. Silloin tieto pysyy varmemmin eheänä kuin ei normalisoidussa tiedossa. Ei-normalisoitua tietoa on kuitenkin nopeampi hakea tietokannasta verrattuna normalisoituun tietoon. Tästä johtuen joissain tapauksissa on perusteltua käyttää ei-normalisoitua rakennetta tiedolle. (Oracle 2002.)

Eräs huomion arvoinen asia on vielä tietokannoissa käytettävät indeksit. Indeksoimalla voidaan nopeuttaa huomattavasti tiedon hakemista tietokannasta. Suuremmissa tietokannoissa indeksit ovat usein välttämättömiä. (Wikipedia 2011.)

Indeksi luodaan yhdelle tietokantataululle kerrallaan. Indeksi asetetaan taulun yhdelle tai useammalle sarakkeelle. Kun tietokannasta haetaan valitun sarakkeen tai sarakkeiden perusteella rajoitetusti tietoa, pystyy tietokanta nopeuttamaan hakua samalla lailla kun sisällysluettelo auttaisi ihmistä etsimään kirjasta haluamansa tiedon. Tietokantaohjelman ei tarvitse käydä koko tietomäärää läpi etsiäkseen osan tiedosta. Tietokanta luo pääavaimelle aina indeksin valmiiksi, jolloin pääavaimen perusteella tiedon etsiminen on nopeaa. (Wikipedia 2011.)

Indeksointi hidastaa jonkin verran tiedon lisäämistä tauluun, koska aina tietoa lisätessä on indeksi päivitettävä uudestaan. Kuitenkin järkevästi asetettu indeksi nopeuttaa tiedonhakua niin paljon, että indeksien käyttäminen on hyödyllistä. (Wikipedia 2011.)

2.2.3 SQL

SQL on relaatiotietokantojen käyttöön tarkoitettu kyselykieli. Sillä voidaan tehdä relaatiotietokannan lukemiseen, kirjoittamiseen ja ylläpitämiseen tarvittavat toimenpiteet. (Wikipedia 2011.)

SQL oli alun perin IBM:n kehittämä kieli. Nykyisin se on standardoitu, jotta samaa kieltä voidaan pääsääntöisesti käyttää eri valmistajien tietokannoissa. (Wikipedia 2011.)

Tietovaraston näkökulmasta hyödyllisimmät SQL-komennot ovat SELECT ja INSERT. SELECT-komennolla tietokannasta voidaan hakea rivejä. INSERT-komennolla rivejä voidaan lisätä. Esimerkin SELECT-lauseesta näkee kuviossa 6, jossa haetaan yhdistetysti tietoa kolmesta tietokantataulusta Myynnit, Tuotteet ja Asiakkaat.

```
-- Haetaan kaikki Matti nimisen asiakkaan tekemät ostot
-- myyntipäivän mukaan järjestettynä
SELECT myyntiaika as Myyntiaika,
  tuote.nimi as Tuote
  m.myyntitulo as Myyntitulo
FROM Myynnit as m, Tuotteet as tuote, Asiakkaat as asiakas
WHERE m.tuote_key = tuote.tuote_key
  and m.asiakas_key = asiakas.asiakas_key
  and asiakas.nimi = "Matti"
ORDER BY m.myyntiaika
```

KUVIO 6. Esimerkki SQL-select lauseesta

Muunlaisissa tietokannoissa myös UPDATE- ja DELETE-komennot ovat yleisiä. UPDATE-komennolla olemassa olevia rivejä voidaan muuttaa. DELETE-komennolla olemassa olevia rivejä voi poistaa.

Tietokantatauluja voi SQL-kielellä luoda CREATE TABLE -komennolla. Taulujen poistaminen tapahtuu DROP TABLE -komennolla. Indeksin tekeminen olemassa olevaan tauluun tapahtuu CREATE INDEX -komennolla. (Wikipedia 2011.)

Lisäksi on monia muita välttämättömiä komentoja. Eri tietokantavalmistajat ovat myös kehitelleet omia SQL-komentojaan, jotka eivät kuulu standardiin. (Wikipedia 2011.)

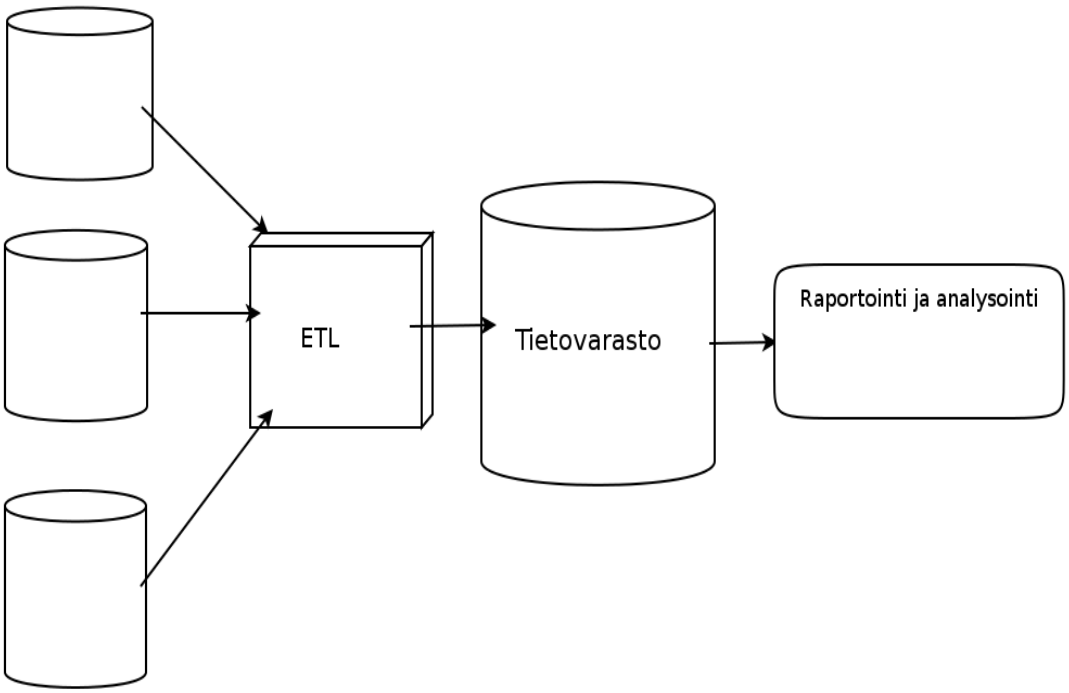
SQL-kielellä ei voi tehdä kokonaisia ohjelmia, joten yleensä SQL:ää käytetään jonkin muun kielen kanssa. Tällöin SQL:llä tehdään kaikki tietokantatoimenpiteet, eli lukeminen ja kirjoittaminen, ja toisella kielellä ohjelman muu toiminnallisuus.

2.2.4 Tietovarastoympäristö

Tietovarasto on tietokanta, joka sisältää päätöksentekoa varten käytettävää tietoa. Tietovaraston tieto ei koskaan katoa tai muutu. Tietovarastoon voi lisätä uutta tietoa, mutta siellä jo olevaa tietoa ei muuteta. Tieto kerätään muista tietokannoista ja integroidaan yhteen tietokantaan, jota kutsutaan tietovarastoksi. Pienemmät tietokannat, joista tieto kerätään, kutsutaan operatiivisiksi kannoiksi. Operatiiviset kannat on tehty tiedon tallennusta silmällä pitäen, kun taas tietovarasto on suunniteltu tiedon lukua ajatellen. Operatiivisen kannan tieto on muuttuvaa, eli tietoa voidaan poistaa tai päivittää. (Inmon, Building The Data Warehouse 3rd Ed 2003, 32.)

Tiedon integroiminen yhteen tietokantaan vaatii usein muutoksia tietoon, koska operatiiviset kannat eivät yleensä ole samanlaisia rakenteeltaan. Esimerkiksi eri tietokantojen tiedot voivat olla tallennettuna eri yksiköihin, jolloin on tehtävä tyyppimuunnokset siirrettäessä tietoa tietovarastoon. Tietojen integroimista kutsutaan myös ETL-vaiheeksi (extract, transform, load), koska siinä ensin ladataan operatiivisen kannan tieto, minkä jälkeen sitä muutetaan, minkä jälkeen se ladataan tietovarastoon. Tietovarastoon tallennetun tiedon on oltava yhtenäistä. (Inmon, Building The Data Warehouse 3rd Ed 2003, 32.)

Tietovaraston tietoa ei päivitetä jatkuvasti. Sen sijaan tieto ladataan tietyin väliajoin, esimerkiksi kerran vuorokaudessa, operatiivisista lähdekannoista(Inmon, Building The Data Warehouse 3rd Ed 2003, 35.). Kuvio 7 sisältää kaavion tietovaraston arkkitehtuurista.



KUVIO 7. Kaavio tietovaraston arkkitehtuurista

2.2.5 Tietovaraston rakentaminen

Tietovaraston rakennetta on hankala suunnitella ennen käyttöä. Tietovaraston vaatimukset saadaan yleensä selville vasta kun tietovarastossa on jo talletettua tietoa ja tietoa on käytetty. Aluksi tietovarastoon tuodaan jokin tietomäärä, jonka jälkeen tietoa jalostetaan ja tehdään käyttäjälle luettavaksi. Kun käyttäjä on käyttänyt tietoa, tehdään tietovarastoon muutoksia ja lisäyksiä palautteen perusteella. Tätä muutosten, lisäysten ja palautteen ketjua kestää koko tietovaraston elinkaaren ajan. Jatkuva muutos myös tuhoaa helposti alkuperäisen suunnittelun mukaisen rakenteen tietovarastosta, jolloin pitkällekin viety suunnittelu voi jäädä turhaksi. (Inmon, Building The Data Warehouse 3rd Ed 2003, 81.)

Tietovaraston tieto on alun perin lähtöisin operatiivisista tietokannoista. Operatiivisten tietokantojen sisältämä tieto on sidoksissa niihin sovelluksiin, jotka tiedon operatiiviseen kantaa lisäävät. Tieto on eri lailla tallennettu eri operatiivisiin tietokantoihin. Usein tallen-

nukseen käytetyt yksiköt ovat erilaisia ja vähintäänkin tietokantojen taulujen ja kenttien nimet ovat erilaiset. Tämän vuoksi tiedon lataaminen useasta operatiivisesta tietokannasta yhteen tietovarastoon voi olla suurikin ongelma. Esimerkiksi yhdessä kannassa voi sukupuolen määrittää numerot 0 ja 1, mutta toisessa kannassa samaa asiaa voidaan ilmaista kirjaimilla n ja m. (Inmon, Building The Data Warehouse 3rd Ed 2003, 83.)

Erinimiset kentät ja eri yksiköt tietokantatauluissa yhtenäistetään jokaista operatiivista tietokantaa kohden tehdyllä kartoituksilla ja muunnoksilla. (Inmon, Building The Data Warehouse 3rd Ed 2003, 85.)

Eräs ongelma on myös se, että operatiiviset kannat voivat olla toteutettuna eri tekniikoilla. Esimerkiksi yhden järjestelmät tiedot voivat olla talletettuna IMS-tietokantaan (IBM Information Management System), toisen DB2-tietokantaan ja kolmannen VSAM-tietokantaan (Virtual Storage Access Method). Eri kannoista tiedon hakeminen on aina erilaista. Helppo ratkaisu on ottaa eri tietokannoista tiedot ulos saman rakenteen omaaviin tiedostoihin, jotka sitten ladataan tietovarastoon. Esimerkiksi yksi tällainen tiedostotyyppi on csv, eli pilkuilla erotetut arvot sisältävä tiedosto. (Inmon, Building The Data Warehouse 3rd Ed 2003, 83.)

Näitä ongelmia tiedon yhtenäistämisestä ei yksinään ole kovin monimutkaista ratkaista, mutta kun ongelmia kasaantuu satoja jokaista operatiivista tietokantaa kohden, ei niiden ratkaiseminen enää olekaan kovin helppoa. (Inmon, Building The Data Warehouse 3rd Ed 2003, 84.)

Tietovaraston tiedon rakenne suunnitellaan kolmessa osassa. Ensimmäinen vaihe on suunnitella tietokannan rakenne korkealla tasolla. Eräs kuvausmenetelmä korkean tason rakenteelle on ER-kaavio. ER-kaavioita on käytetty esimerkiksi tämän opinnäytetyön kuvioissa 4 ja 5. ER-kaavio kuvaa tietokokonaisuuksien väliset suhteet. (Inmon, Building The Data Warehouse 3rd Ed 2003, 94.)

Seuraava vaihe on määritellä yksittäiset ensimmäisen vaiheen ER-kaaviossa kuvatut tietueet. Tarvittaessa tietueita pilkotaan useammaksi tauluksi. Tauluille määritellään tietueisiin tietoja yhdistävät relaatiot ja ryhmittelyt käyttäen avainkenttiä. Myös tietojen tyypit määritellään. (Inmon, Building The Data Warehouse 3rd Ed 2003, 95.)

Kaikenkaikkiaan tietovarasto kannattaa toteuttaa vähän kerrallaan iteraatioissa. Suurimmat hyödyt tästä on, että näkyviä tuloksia saadaan heti aikaan, ja että käyttäjän palautetta saadaan nopeasti ohjaamaan kehityksen suuntaa. (Inmon, Building The Data Warehouse 3rd Ed 2003, 102.)

Tietovarastoon tallennetaan myös metatietoa, eli tietoa tiedosta. Metatieto on tietovaraston toteuttajan rakentama tietorakenne, joka kertoo miten varsinainen tieto muodostuu. Käyttäjä hyötyy metatiedosta helpomman tiedonlöytämisen myötä, koska tietoa pystytään esittämään selkeämmin. (Inmon, Building The Data Warehouse 3rd Ed 2003, 113.)

Metatieto on oikeastaan kaikki informaatio tiedossa, mikä ei itsessään ole tietoa. Metatieto on tietoa tiedosta, eli se voi kertoa tiedon alkuperän tai vaikkapa tarkoituksen. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 14.)

Tietovaraston sisältää kaiken oleellisen tiedon, jota operatiivisista kannoista saa. Tietoa voi käyttää sellaisenaan, mutta yleisempää on analysoida tietoa jollakin tietokoneohjelmalla, jonka tuottamaa tietoa sitten tarkastellaan. (Inmon, Building The Data Warehouse 3rd Ed 2003, 136.)

Jos tietovaraston suorituskykyä halutaan parantaa monimutkaisempien tietokantakyselyjen kohdalla, jotka vaikkapa hakevat useasta taulusta tietoa, tai suorittavat raskaita laskutoimenpiteitä, voidaan käyttää summataulua. Summatauluun kerätään tietoa valmiiksi sellaiseen muotoon, jota käytetään usein, mutta jota ei suoraan saa tietokannasta. Summataulu päivitetään aina, kun tietovarastonkin tieto päivitetään, jolloin se pysyy ajan tasalla. Esimerkiksi, jos monimutkainen tietokantakysely, joka suorittaa laskutoimenpi-

teitä tiedolle ja hakee tietoa useasta eri tietokantataulusta, kestää kymmenen minuuttia, niin voidaan luoda samantyylisen tiedon sisältävä summataulu. Tähän summatauluun on valmiiksi laskettu tietokantakyselyllä kerätty tieto. Tästä summataulusta tietokantakyselyn suorittaminen voisi kestää vain sekunteja tai millisekunteja.

2.2.6 Tietovaraston tekniikka

Tietovarastojen käyttämä tila voi olla hyvin suuri. Normaaleissa tietokannoissa tietokannan viemä koko mitataan megatavuissa tai gigatavuissa. Suuremmissa tietovarastoissa koko mitataan jo teratavuissa tai petatavuissa. (Inmon, Building The Data Warehouse 3rd Ed 2003, 167.)

Tämä tiedon suuri määrä johtuu tietovaraston sisältämän tiedon tarkkuudesta ja pitkäikäisyydestä. Tietovarastojen tieto tallennetaan yleensä mahdollisimman tarkasti, eikä vanhaa dataa poisteta. (Inmon, Building The Data Warehouse 3rd Ed 2003, 167.)

Tietovaraston suuri koko aiheuttaa vaatimuksia käytettävälle teknologialle. Tietovarastosta pitää saada nopeasti tietoa ulos, vaikka koko kasvaisikin suureksi. (Inmon, Building The Data Warehouse 3rd Ed 2003, 168.)

Tietovarastoon tallennettu tieto pitää pystyä indeksoimaan, jotta suuresta datamäärästä saa nopeasti haettua haluamansa. Tämä rajoittaa pois tietyntyyppiset massamuistit, kuten nauha-asemat. Käytännössä tämä tarkoittaa kovalevyjen käyttöä. Tieto myös kannattaa jakaa usealle fyysiselle levylle, jotta tietoa voi hakea monesta levystä yhtä aikaan. (Inmon, Building The Data Warehouse 3rd Ed 2003, 170-171.)

Tietovarastoon pitää saada liikutettua helposti tietoa sisään operatiivisista kannoista. Myös tiedon liikkuminen ulos kannasta pitää olla helppoa. Tiedon hakemiselle ja lisäämiselle pitää siis olla helposti käytettävissä oleva rajapinta. (Inmon, Building The Data Warehouse 3rd Ed 2003, 170.)

Tietovarastoja varten on kehitetty tietokantaohjelmia, joiden tarkoituksena on toimia tietokantana tietovarastoa silmällä pitäen. Näissä tietokannoissa on keskitytty vain tietovarastolle oleellisiin ominaisuuksiin, ja joitain yleiskäyttöisten tietokantojen ominaisuuksia on jätetty pois. Esimerkiksi muutosten peruminen (englanniksi rollback) on yleinen ominaisuus yleiskäyttöisissä tietokannoissa, mutta tietovarastossa se on turha ominaisuus. Tämä ominaisuus voi jopa hidastaa tietokannan käyttöä, vaikka ominaisuutta ei käyttäisikään, sillä muutosten kirjaaminen aiheuttaa tietokantaohjelmalle paljon ylimääräistä toimintaa. (Inmon, Building The Data Warehouse 3rd Ed 2003, 180.)

Esimerkkejä tietovarastointiin soveltuvista tietokantaohjelmistoista ovat DB2, Informatix, Microsoft SQL Server ja Oracle Database. Nämä ovat tosin soveltuvia myös yleiskäyttöisiksi tietokannoiksi.

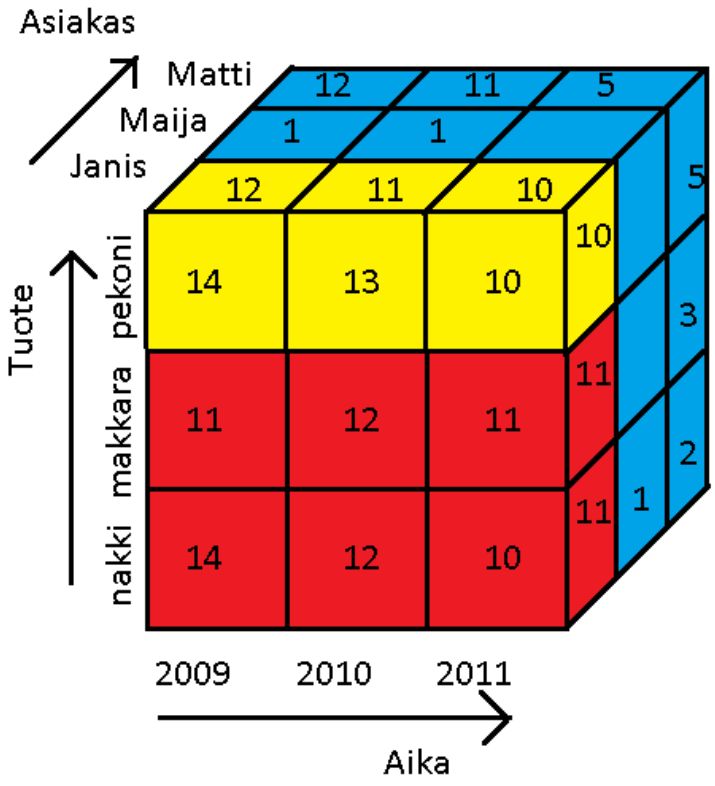
2.2.7 Tiedon esittäminen

Yksinkertaisin tapa tietovaraston tiedon esittämiseen on luoda staattisia raportteja, joita käyttäjät lukevat. Tällaisissa raporteissa käyttäjällä ei ole kovin paljon mahdollisuutta valita mitä tietoa katsotaan, mutta toisaalta yksinkertaiset raportit eivät vaadi minkään työkalun opettelua. Käyttäjälle hieman vaikeampi, mutta antoisampi tapa on käyttää jotain tiedonhakutyökalua. Nämä vaativat jonkun verran koulutusta, jotta käyttö luonnistuu, mutta toisaalta käyttäjä voi käytännössä aina löytää haluamansa tiedon. Viimeinen vaihtoehto olisi käyttää suoraa tietokantakyselyä, mutta harva tuntee kyselykieltä riittävän hyvin, eikä tietoa saa helposti havainnolliseen muotoon edes osaava käyttäjä.

Tietovaraston tiedon esittäminen järkevästi käyttäjille on tärkeää tietovarastoa hyödyntäessä. Tieto esitetään dimensionaalisessa muodossa käyttäjälle. Tämä vaatii käytännössä sen, että tietovaraston tietorakenne on myös tässä muodossa. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 10.)

Dimensionaalisella tietorakenteella tarkoitetaan sellaista rakennetta, jossa tieto esitetään mittareilla ja dimensioilla. Mittari tarkoittaa mitattavaa asiaa, kuten vaikkapa myyntitulo- ja. Dimensiot ovat mitattavan asian sijaintia määrittävät muuttujat, kuten vaikkapa aika, sijainti asiakas ja tuote. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 11.)

Tällaista tietorakennetta voisi ajatella kuutiona, jossa voi tosin olla enemmän kuin kolme ulottuvuutta. Kuvio 8 havainnollistaa tiedon esittämistä kuutiolla. Kuvion dimensiot ovat tuote, asiakas ja aika. Numerot ovat faktoja, ja voivat tarkoittaa vaikkapa myyntituloja. Kuution reunat on nimetty dimension mukaan. Kuutiosta voi sitten tarkistella yksittäistä osaa tai suurempaa osaa rajoittamalla joitain dimensioita joko yksittäiseen arvoon tai suurempaan arvojoukkoon. Esimerkiksi kuvion 8 kuution keltainen alue sisältää kaikki tiedon Janis-asiakkaan ostamasta pekonista vuosilta 2009 - 2011 ja punainen kaikki saman asiakkaan makkara- ja nakkiostokset. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 10.)



KUVIO 8. OLAP-kuutio

Esimerkiksi myyntituloja voisi tarkastella rajoittamalla sijainnin Helsinkiin ja ajan vuodelle 2011. Näin näkisi Helsingissä vuonna 2011 tapahtuneet myyntitapahtumat. Aikaa voisi myös rajoittaa tarkemmalla tasolla, kuten kuukausi tai päivätasolla, tai rajauksen voisi ottaa kokonaan pois. Tätä ideaa käyttäen käyttäjä saa yksinkertaisesti haluamansa tiedon esille. Tiedon löytäminen helpottuu myös kun dimensiot jaetaan eri tasoihin. Esimerkiksi ajan voi jakaa vuosiin, vuosineljänniksiin ja kuukausiin. Sijainnin voisi jakaa maahan, kaupunkiin ja osoitteeseen.

Tietoon voidaan porautua aloittamalla tapahtumien tarkastelu korkeimmalla dimensioiden tasolla. Tästä poraudutaan alaspäin rajoittamalla ensin yhtä dimensiota alemmalle tasolle, jolloin dimension sisältämien arvojen joukko pienenee. Tätä voi jatkaa, kunnes on porauduttu kaikilla dimensioilla alimmalle tasolle, eli tasolle, joka kertoo yksittäisen tapahtuman tasolla asian.

Dimensionaalinen tietorakenne vaatii usein, että tieto toistuu tietokannassa useaan kertaan (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 11). Tämä johtuu siitä, että pitääkseen tiedonhaun yksinkertaisena ja nopeana suositaan usein tähtimallia tietokannalle. Lumihiutalemallista aiheutuvat ongelmat tiedonhaussa ovat lähinnä monimutkaiset tietokantakyselyt ja tiedon haun hitaus.

Tiedon esittämistä dimensionaalisessa muodossa kutsutaan OLAP:ksi (online analytic processing) (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 13).

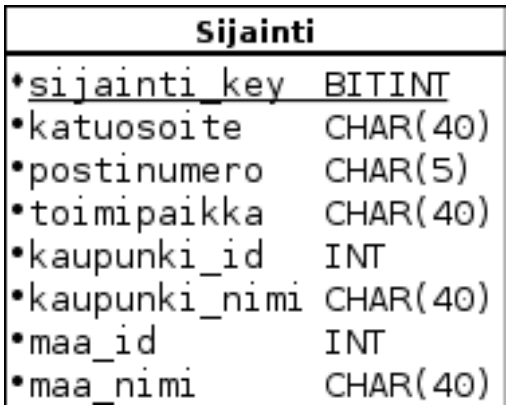
2.2.8 Faktat ja dimensiot

Kun tietoa kuvataan mittareilla ja dimensioilla, on alla olevan tietorakenteen hyvä olla samanlainen. Tieto jaetaan fakta ja dimensiotauluihin.

Faktataulu on tärkeämpi kuin dimensiotaulu siinä mielessä, että faktataulu sisältää varsinaisen mittari-informaation, kuten esimerkiksi myyntitapahtuman (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 17). Faktataulu sisältää jokaista tapahtumaa kohden yhden rivin. Faktataulu sisältää aina varsinaiset arvot tapahtumalle sekä avaimia viitattaviin dimensiotauluihin. Lisäksi faktataulussa on yleensä pääavain, joka koostuu dimensioviittauksista. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 19.)

Faktan arvo on järkevintä olla jokin numeerinen arvo, jolloin sitä voi summata korkeammalle tasolle, eli kun käyttäjä porautuu tiedossa alaspäin. Yleisin arvo on jokin rahavaluutta. Jos arvona on jokin muu kuin numeerinen arvo, ei tietoa voi helposti summata ylöspäin. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 17.)

Dimensiotaulu sisältää tekstiselityksen faktataulun dimensioavaimelle (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 19). Esimerkiksi sijainnin avain voisi olla S001 ja tekstiselitteenä avainta vastaava katuosoite. Dimensiotaulu voi sisältää myös muuta tietoa, kuten tiedot tasojen hierarkiasta ja viittauksia muihin dimensioihin ja muuta tekstuaalista informaatiota. Kuvio 9 sisältää esimerkin sijaintidimensiosta, jossa ovat osoittetason lisäksi tasot kaupunki ja maa.



KUVIO 9. Esimerkki sijaintidimensiotaulusta

Dimensiotaulut ovat välttämättömiä tiedon löytämisen ja ryhmittelyn kannalta. Dimensioiden kuuluu sisältää kaikki tekstiarvot, joita käyttäjä näkee käyttäessään tiedonhaku ohjelmaa. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 19.)

Faktatauluun yhdistetään dimensiot avainten perusteella. Yksi faktataulun rivi yhdistettynä dimensiotaulujen tietoon antaa tarkan kuvauksen tapahtumasta. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 22.)

Faktoihin ja dimensioihin perustuva tietorakenne on hyvin skaalautuva. Uuden tapahtuman kirjaaminen onnistuu lisäämällä yksi rivi faktatauluun. Tarvittaessa dimensiotauluihin on myös lisättävä rivi, jos vastaavanlaista tapahtumaa ei ole tietovarastoon laitettu. Uusia tarpeita voi helposti paikata joko lisäämällä rivejä tauluun, tai muuntamalla taulujen rakennetta lisäämällä niihin uusia kenttiä esimerkiksi SQL alter table -komennolla. (Kimball & Ross, The Data Warehouse Toolkit 2nd Ed 2002, 25.)

2.3 Visual Studio Tools for Office

Käyttämällä Microsoft Officea sovelluksen päällimmäisenä kerroksena saadaan käyttäjälle tuttu käyttöliittymä. Lisäksi saadaan Microsoft Wordin tekstinkäsittelyominaisuudet, Microsoft Excelin tiedonanalysointiominaisuudet ja Microsoft Outlookin sähköpostitilien käsittelyominaisuudet. Tällaisia sovelluksia voi kehittää Microsoft Visual Studiolla käyttäen Visual Studio Tools for Office (VSTO) -työkalua. Nämä sovellukset lisäävät Office-ohjelmistoon tiettyyn tarkoitukseen kohdistettua toiminnallisuutta. Esimerkiksi Excelin toiminnallisuutta voisi laajentaa lisäämällä automaattisen tietojenhaun tietokannasta ja automaattisen tallentamisen tietokantaan. Office ei tarvitse verkkoyhteyttä, joten tämä on käytännöllisempi tapa verrattuna Web-pohjaiseen arkkitehtuuriin. (Microsoft, 2011.)

Visual Studiolla voi kehittää kahdenlaisia Office-sovelluksia, asiakirjatason laajennuksia ja sovellustason laajennuksia. Asiakirjatason laajennukset toimivat tietyn dokumentin tai dokumenttipohjan yhteydessä. Sovellustason laajennukset toimivat koko Office-

ohjelmiston yhteydessä. Jos halutaan liittää esimerkiksi johonkin Excel-taulukkoon lisää ominaisuuksia, tehdään asiakirjatason laajennus. Jos taas esimerkiksi halutaan tehdä oma valikko, joka on kaikissa dokumenteissa läsnä, tehdään sovellustason laajennus. (Microsoft, 2011.)

Asiakirjatason laajennus sisältää ohjelmakoodin, joka ajetaan tietyn dokumentin tai dokumenttipohjan yhteydessä. Ohjelmoitu toiminnallisuus on käytettävissä vain silloin, kun dokumentti on auki. Asiakirjatason laajennuksiin voi liittää myös Microsoft Forms -toiminnallisuutta, jolla saadaan lisättyä vaikkapa dokumenttiin liittyviä dialogi-ikkunoita. (Microsoft, 2011.)

Sovellustason laajennukset lisäävät toiminnallisuutta koko Office-ohjelmistoon. Näitä laajennuksia voi käyttää minkä tahansa dokumentin yhteydessä. (Microsoft, 2011.)

2.3.1 Vertailu muihin tekniikoihin

Office-ohjelmistoon voi tehdä automatisointia myös Microsoft Visual Basic for Applications (VBA) -kielellä. Visual Studiolla tehtyssä VSTO-projektissa on etuna .NET-Framework -ohjelmistokehys (msdn.microsoft.com, 2011.). Microsoft Officen ohjelmointi ei VSTO:n ja VBA:n lisäksi onnistu muilla ohjelmointityökaluilla (Microsoft, 2011.).

VBA:lla ohjelmoitu koodi tallentuu suoraan dokumenttiin. VSTO-laajennuksen koodi puolestaan tallennetaan erilliseen DLL-tiedostoon. (Microsoft, 2011.)

VBA-makrot pystyvät käyttämään Officen komponentteja, ja käytössä on myös VBA-rajapinnat. VSTO:lla on käytössä paremmat .NET Framework -rajapinnat. (Microsoft, 2011.)

VBA on hyvin yksinkertainen kieli, joka on tarkoitettu lähinnä helpoksi kirjoittaa. VSTO sopii paremmin suurempiin projekteihin, jotka tarvitsevat enemmän toiminnallisuutta. (Microsoft, 2011.)

Kaikenkaikkiaan Visual Studio tarjoaa paremman kehitysympäristön. Käyttäjän näkökulmasta pienissä sovelluksissa eroa ei huomaa, mutta suuremmissa sovelluksissa VBA:lla tehty ohjelma voi toimia hitaasti. Hyvin pienissä projekteissa VBA:lla on etu yksinkertaisuuden ja vaivattoman käyttöönoton takia. Suuremmissa projekteissa Visual Studion edut tulevat esille. (Microsoft, 2011.)

2.3.2 Käyttäminen

Kun käyttäjä avaa sovellukseen liittyvän dokumentin, Office lataa VSTOLoader.dll kirjaston, joka puolestaan lataa .NET kirjaston ja käynnistää laajennoksen sisältämän ohjelman. Jos dokumentti ei ole paikalliselta koneelta, vaan esimerkiksi verkkopalvelusta, tarkistaa Office, onko dokumentti luotetussa sijainnissa. Jos dokumenttia ei todeta turvalliseksi, laajennusten lataaminen keskeytetään. Tässä vaiheessa Office voi myös tarkistaa, onko laajennukselle saatavilla päivityksiä, jotka ladataan tarvittaessa. (Microsoft, 2011.)

Office-ohjelmien kehittäminen Visual Studiolla vaatii Professional version ohjelmasta, sekä Microsoft Office 2003:n tai uudemman version asennuksen kehityskoneella. Lisäksi asennettuna pitää olla Visual Studio Tools for Office -ajoympäristö ja .NET ajoympäristö. Visual Studiolla kehitettyjen Office-ohjelmien ajaminen vaatii loppukäyttäjältä Office-asennuksen sekä VSTO-ajoympäristön ja .NET -ajoympäristön. (Microsoft, 2011.)

Ohjelman jakelemisen loppukäyttäjälle voi tehdä Visual Studion rakentamalla asennusohjelmalla. Asennusohjelma asentaa sovelluksen tarvitseman dokumentin sekä ohjelman

sisältävän kirjaston. VSTO ajoympäristö ja .NET ajoympäristö käyttäjän on asennettava itse.

Dokumenttitason sovelluksissa jakelun voi hoitaa myös jakelemalla ohjelman binäärit erillisinä tiedostoina, jolloin käyttäjän ei tarvitse ajaa asennusohjelmaa, mutta tällöin on käyttäjän kuitenkin huolehdittava, että tiedostot ovat samassa kansiossa. (Microsoft, 2011.)

3 ENNUSTESOVELLUS

Projektina on tehdä Logican asiakkaan tilaama uusi proof of concept (POC) -versio vanhasta ennustussovelluksesta Microsoft Excelin päälle tehtynä. Edellinen järjestelmä on web-pohjainen sovellus, joka halutaan korvata uudella hitaan ja epävarman toiminnallisuuden takia..

Proof of concept -version on tarkoitus osoittaa, että sovelluksen on mahdollista toimia myös Excelin päälle tehtynä. Exceliä halutaan käyttää, koska se on kohderyhmälle jo tuttu työväline, joten koulutusta sovelluksen käyttöön ei tarvitse tehdä niin paljon. Ohjelman on tarkoitus toimia vain demona, eikä täyttä toiminnallisuutta vielä tässä vaiheessa vaadita.

Excelin etuna web-sovellukseen nähden on myös se, että tietoja voidaan tarkastella ilman verkkoyhteyttä, kunhan ne vain on ensin ladattu kertaalleen. Esimerkiksi talouspuolen henkilöt voivat ladata tietokannasta tiedot kannettavaan tietokoneeseensa ja sitten esitellä niitä toisille ihmisille muualla ilman verkkoyhteyttä.

Sovelluksen täytyy pystyä hakemaan käyttäjän valitsemat tiedot tietokannasta ja esittämään ne järkevästi käyttäjälle. Käyttäjän täytyy voida tehdä muutoksia haluamiinsa tietoihin, minkä jälkeen tiedot pitää pystyä tallentamaan tietokantaan.

Tiedot, joita sovelluksella käsitellään, ovat ennustetietoja. Ennustetiedot ovat markkina-alue-, tuote- ja aikakohtaisia. Yksi ennuste käsittelee kerrallaan yhtä markkina-aluetta, joten tietojen esittämisessä täytyy ottaa huomioon aika- ja tuotedimensiot.

3.1 Olemassaolevan tietokannan käyttäminen

Uutta tietokantaa ei POC-versiota varten ole tarkoitusta tehdä, vaan käytetään ennemmin jo olemassa olevaa tietokantaa. Osa työstä on vanhan tietokannan rakenteen selvittämistä.

Käytössä oleva tietokanta on Windows-palvelimelle asennettu Oracle Database versiolla 10.2. Tietokanta on kehityskäytössä, ja varsinainen tuotantokäytössä oleva tietokanta on asiakkaan palvelimella.

Tietokannan taulut koostuvat dimensiotauluista, faktatauluista ja järjestelmän sisäiseen käyttöön tarkoitetuista tauluissa. Fakta- ja dimensiotaulut sisältävät kaiken ulkopuolelta tuodun tiedon ja käyttäjien järjestelmään syöttämän tiedon.

Tietokannan rakenne on alun perin suunniteltu monikäyttöiseksi. Tässä on ollut ideana se, että samaa järjestelmää olisi ollut mahdollisimman pienellä vaivalla käyttää muuallakin.

Tietokannan monikäyttöisyys on käytännössä tehty tallentamalla eri dimensioiden ja faktojen tiedot tietokantariveinä asetustauluihin sen sijaan, että ne oltaisiin tehty konkreettisina tauluina tietokantaan. Tästä seuraa ikävä kyllä tietokantakyselyjen monimutkaistuminen ja hidastuminen. Lisäksi rakennetta on hankalampi ymmärtää, koska taulut on tehty yleiskäyttöisiksi, ja niiden nimeäminen on myös yleiskäyttöistä.

Sen sijaan, että dimensioille oltaisiin kullekin tehty omat taulunsa, on taulu DIM_INT_DIMENSIONS, joka sisältää eri dimensioiden tiedot. Varsinaiset dimensiorivit ovat DIM_INT_DIMENSION_VALUES taulussa, jossa on siis dimension tyyppi ja varsinaiset tiedot. Kaikkien dimensioiden rakenne on näin ollen samanlainen.

Faktojen kanssa on tietokanta samantyylinen kuin dimensioidenkin, eli faktojen määritykset ovat yhdessä taulussa ja arvot toisessa. Todennäköisesti tämä tietokantarakenne

on osasyy vanhan sovelluksen heikolle toiminnallisuudelle ja hidastelulle. Kannan rakenne ei ole kovin ystävällinen kehittäjän näkökulmasta, eikä myöskään tehokkuusnäkökulmasta.

Tietokantayhteys .NET-sovelluksesta Oraclen tietokantaan ei ole aivan itsestään selvä asia. Koska .NET-alustassa ei ole valmista kirjastoa Oracle Database tietokannan kanssa toimimiseen, on käytettävä Oraclen omaa kirjastoa. Onneksi Oraclen kirjastot on tehty myös .NET-kielille.

3.1.1 Tietokantayhteys

Oraclen ohjelmakirjastojen käyttöön ottamiseksi täytyy joko asentaa Oraclen asiakasohjelma tai hakea Instant Client ohjelma Oracle web-sivuilta. Oraclen asiakasohjelma on noin 700 megatavun kokoinen asennuspaketti, joka on turhan suuri yksinkertaisen tietokantayhteyden saamiseksi.

Instant Client sen sijaan on pelkät Oraclen tietokantakirjastot sisältävä paketti, josta voi liittää ohjelmaansa vain haluamansa kirjastot. Instant Clientistä tulee ylimääräistä kokoa ohjelmaan noin 17 megatavua, mikä on jo hyväksyttävissä oleva koko.

Oracle Instant Client kirjaston saa käyttöön .NET-ohjelmassa kopioimalla tiedostot oraociicus10.dll, oci.dll, orannzsbb10.dll ja oraoccil10.dll Instant Client kansiosta samaan kansioon sovelluksen kanssa. Tämän jälkeen lisätään projektiin kirjasto viittaus System.Data.OracleClient.dll, joka on Microsoftin tekemä kirjasto, joka puolestaan hyödyntää Oraclen kirjastoja toiminnallisuuteen. Jotta kirjastoa voi käyttää, täytyy ohjelmakoodiin lisätä kuvion 10 mukainen rivi.

```
using System.Data.OracleClient;
```

KUVIO 10. Ohjelmakoodi Oracle tietokannan käyttöönottoa varten.

Yhteyden muodostamiseksi tarvitaan vielä Oraclen connection string, joka on merkkijono, joka ilmaisee yhdistämiseen tarvittavat tiedot. Esimerkki connection stringistä on kuviossa 11.

```
palvelimen.osoite.fi = (DESCRIPTION = (ADDRESS =
   (PROTOCOL = TCP) (HOST = palvelimen.osoite.fi)
   (PORT = 1234))
  (CONNECT_DATA = (SID = oraclesid1)) )
```

KUVIO 11. Connection string esimerkki

Tämä kuitenkin vaatii, että samassa kansiossa ohjelman kanssa on tiedosto tnsnames.ora, jossa määritellään oraclesid1-kohteelle tarkemmat tiedot. Tämän voi välttää käyttämällä pidempää versiota connection stringistä, joka sisältää myös SID-tiedon. Esimerkki tällaisesta connection stringistä on esitelty kuviossa 12.

```
USER ID = tunnus;PASSWORD=salasana;DATA SOURCE =
  (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)
    (HOST = palvelimen.osoite.fi)(PORT = 1234))
  (CONNECT_DATA = (SID = oraclesid1)) )
```

KUVIO 12. Vaihtoehtoinen connection string

Kun connection string on määritelty, voidaan yhteys Oraclen tietokantaan ottaa helposti. Kuvio 13 antaa esimerkin yhteyden tekemiseksi ja tietokantakyselyn tekemiseksi C#-kielellä. Kuvion koodi luo yhteyden tietokantaan connectionString-muuttujaan asetetun merkkijonon perusteella. Tämän jälkeen luodaan SQL-komento, joka sitten ajetaan kutsumalla ExecuteReader-metodia, joka palauttaa OracleDataReader-olion. Select-komennon palauttamat tietokantarivit luetaan kutsumalla OracleDataReader-olion Read-metodia jokaiselle riville. Kuvion koodi tulostaa luetuista riveistä nimi-kentän konsoliin.

```
static private void connectAndQuery
{
    string connectionString = getConnectionString();
    using (OracleConnection conn = new OracleConnection())
    {
        conn.ConnectionString = connectionString;
        conn.Open();

        OracleCommand command = conn.CreateCommand();
        string sql =
            @"SELECT *
              FROM Asiakkaat";
        command.CommandText = sql;

        OracleDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            string nimi = reader["nimi"].ToString();
            Console.WriteLine(nimi);
        }
    }
}
```

KUVIO 13. Esimerkki Oracle-tietokantaan yhdistämisestä

3.2 Toimintalogiikka

Sovelluksen tärkein osa on tietojen automaattinen käsittely Excel-taulukossa. Sovellus lataa tietokannasta tuotteet, ja tuotteille eri mittariarvoja halutuille kuukausille. Tuotteiden tulee olla tietokannan määrittelemässä hierarkisessa järjestyksessä.

Tuotedimensio on jaettu useaan eri tasoon. Tietokannasta löytyy tiedot alimmalle tasolle. Ylempien tasojen tiedot täytyy summata alimman tason tiedoista.

Tuotehierarkian muodostuksessa päädyttiin lukemaan ensin tietokannasta kaikki tuotedimensioarvot ja käyttämään .NET kirjaston tietorakenteita sen tallentamiseen. Näin tietokantaa ei rasiteta turhaan, eikä olemassa olevan tietokannan hankala dimensioratkaisu haittaa niin paljon ohjelman toimintaa. Tietorakenteena käytetään .NET kirjaston geneeristä hajautustaulukontaineria eli säiliötä, jonka avaimena toimi tuotteen tunniste ja

arvona linkitetty lista, joka sisältää alituotteiden tunnisteita. Kuvio 14 sisältää tuotehierarkian sisältävän tietorakenteen määrittelyn C#-kielellä.

```
Dictionary< string, LinkedList<string> > productHierarchy;
```

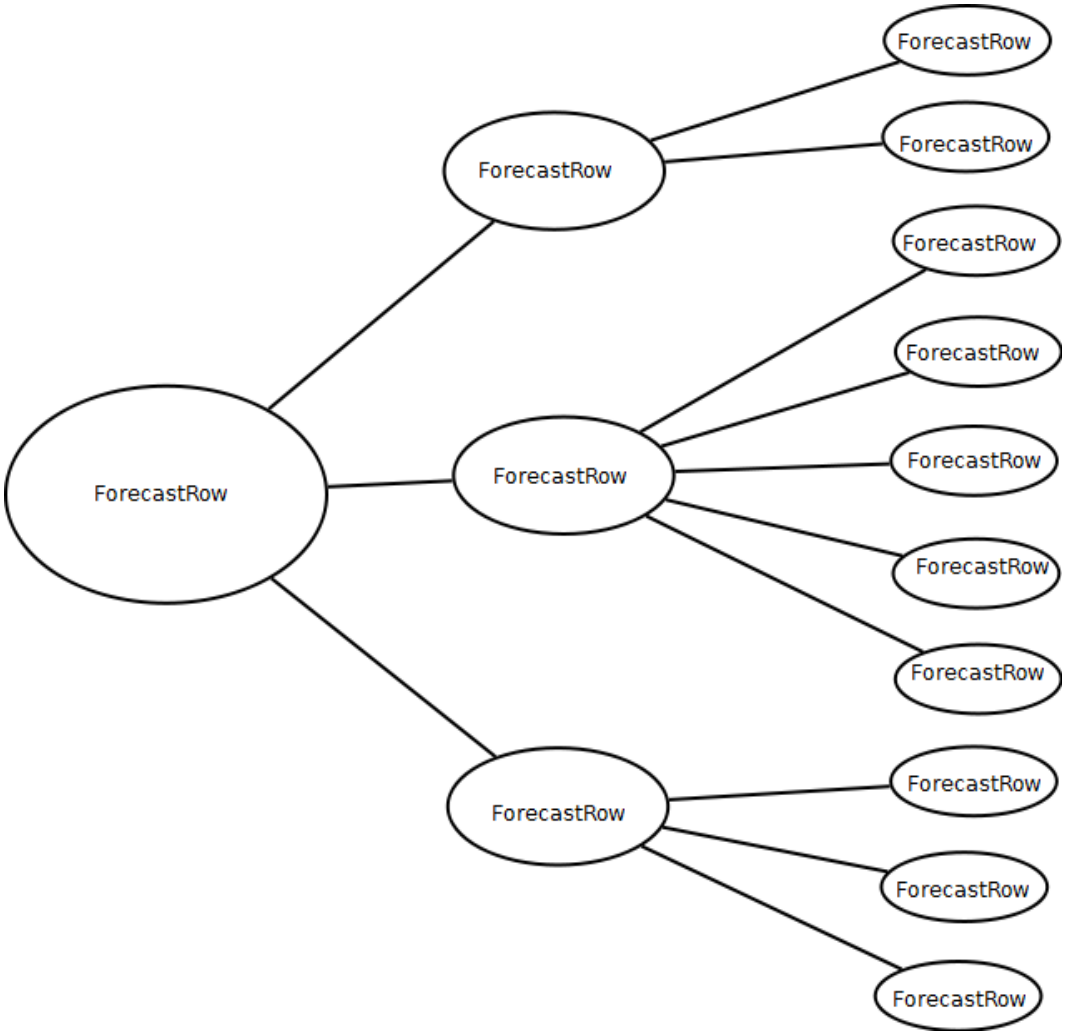
Kuvio 14. Tietorakenne tuotehierarkian varastointia varten

Kun tuotehierarkia on muodostettu, täytyy tietokannasta hakea varsinaiset arvot eli faktat tietokannasta. Faktojen yhdistämiseksi tuotehierarkiaan käytetään rivikohtaista puu tietorakennetta. Jokainen rivi sisältää tiedon lapsiriveistä ja emorivistä sekä tietenkin rivin fakta-arvot. Kuvio 15 esittää puurakenteen lehden attribuutit C#-kielellä. Ohjelma rakentaa puun solmousta koostuvan rakenteen kuviossa 14 määritellyn hajautustaulukontainerin sisällön perusteella.

Tämän tietorakenteen avulla pystyy puurakenteessa helposti liikkumaan sekä ylös että alaspäin, mikä on tarpeen uusien arvojen laskemiseksi nopeasti. Puurakennetta kuvaava graafi on kuviossa 16.

```
class ForecastRow
{
    public string productId;
    public string productName;
    public List<double> forecastValues;
    public ForecastRow parentRow;
    public List<ForecastRow> childRows;
    public int level;
    //...
```

KUVIO 15. Sovelluksessa käytetyn puurakenteen solmun määrittely



KUVIO 16. Ennusteriveistä koostuva puurakenne

Käyttäjä voi muuttaa Excel-taulukon arvoja halutunlaisiksi. Uudet arvot voi käyttäjä halutessaan tallentaa tietokantaan. Jos käyttäjä muuttaa arvoja tuotehierarkian alimmalla tasolla, päivitetään tiedot automaattisesti ylöspäin. Jos käyttäjä muuttaa tietoja ylemmältä tasolta, jaetaan tieto olemassa olevien arvojen perusteella alempiin tuotetasoihin. Samalla tietenkin ylemmät tasot päivittyvät. Ideana on, että hierarkian arvot pysyvät aina oikeana, ja lasketaan käyttäjän syöttämän arvon mukaan, syötettiin tämä arvo sitten mihin tahansa.

Tietoa voi olla kerrallaan muokattavana jopa kymmeniätuhansia rivejä. Tämän tietomäärän hallitseminen nopeasti sujuu helposti .NET kirjaston sisältämillä tietorakenteilla, mutta Excel-ohjelman taulukkoon kirjoitus, ja sieltä luku solu kerrallaan voi kestää minuutte-

ja. Nopeus on kuitenkin tärkeää, sillä yksi vanhemman version suurista ongelmista oli sen huono suorituskyky.

Excelin VSTO-rajapinta antaa mahdollisuuden muokata useampaa solua kerrallaan. Tämä tapahtuu helposti C#-kielen kaksiulotteisten taulujen avulla. Tästä johtuen on tieto kuitenkin ennen käyttäjän näkyville laittamista kopioitava vielä yhteen ylimääräiseen tietorakenteeseen. Tämän myötä soluihin Excel-taulun kirjoittaminen on lähes välitöntä. Esimerkki Excel-tauluun kirjoittamisesta on esitelty kuviossa 17.

```
string table[,] = new string[numRows, numColumns];

// täytä table-taulukko tässä...

Range[startCell, endCell] = table;
```

KUVIO 17. Esimerkki Excel tauluun kirjoittamisesta nopeasti

Ennen kuin tietoja haetaan käyttäjän näkyville, täytyy käyttäjän määritellä mitä tietoja halutaan etsiä. Tämä tapahtuu erillisessä dialogi-ikkunassa. Ikkunassa pystyy määrittelemään käytettävän ennustuksen ja halutun aikarajauksen sekä ennustettavat mittarit.

Kyselydialogi on toteutettu Windows Forms -tekniikalla, jota on helppo käyttää Excel-sovelluksen yhtedessä. Ikkunan ulkoasun tekeminen onnistuu graafisilla työkaluilla, jolloin varsinaista ohjelmointityötä muodostuu vain toiminnallisuuden toteuttamisesta.

Dialogin avautuessa haetaan tietokannasta tiedot siitä, mitä mittareita on olemassa, sekä mitä eri ennustuksia on luotu. Tämän jälkeen käyttäjä valitsee haluamansa ennusteen nimen listasta. Kun valinta on tehty, haetaan ennusteelle löytyvät päivämäärät kuukausitasolla tietokannasta käyttäjälle valittavaksi. Käyttäjän valittua haettaville tiedoille alku- ja loppupäivämäärät haetaan varsinaiset tiedot.

Kun tiedot on haettu, voi käyttäjä tehdä haluamansa muutokset niihin. Tämän jälkeen käyttäjä painaa talletusnappia, jolloin tiedot lähetetään tietokantaan.

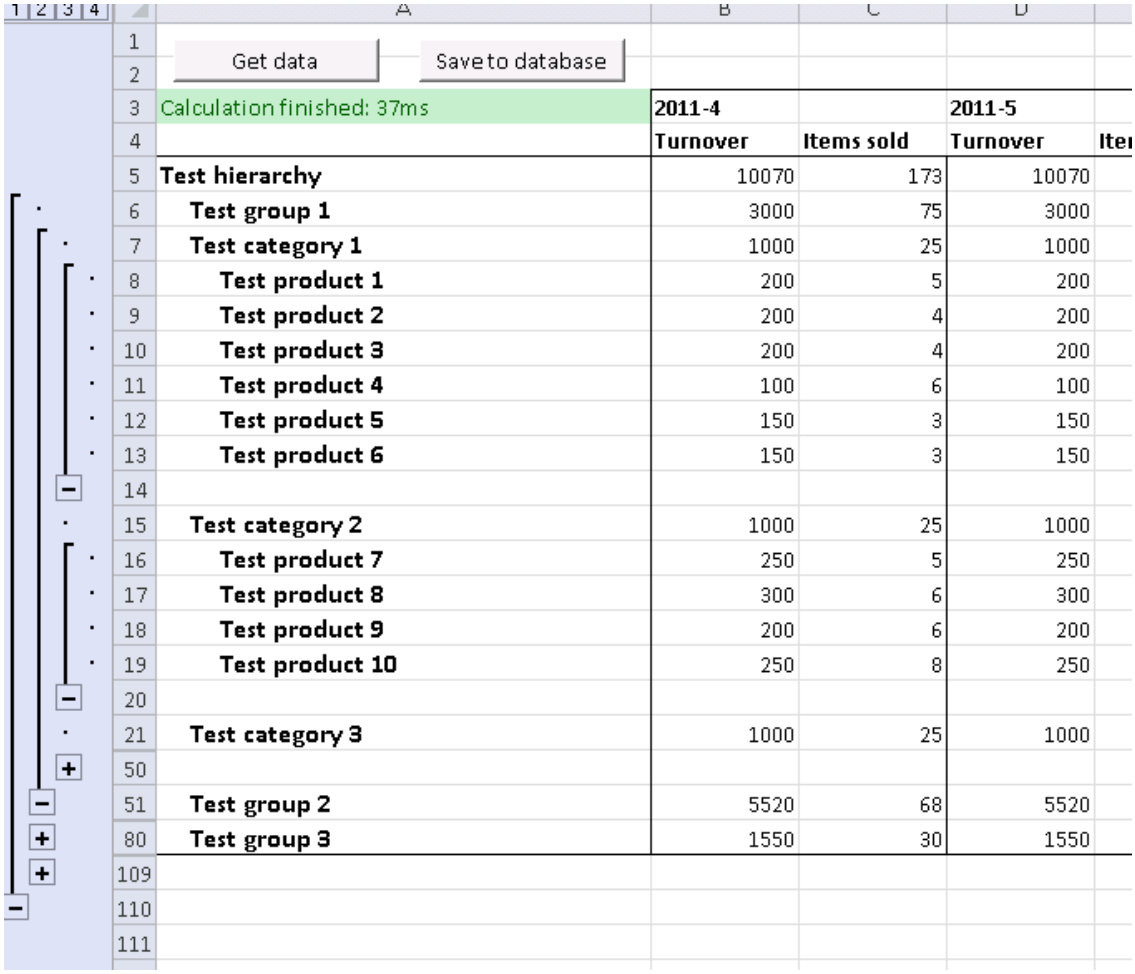
Tietoja tallentaessa tarkistetaan, mikä Excel-taulun soluista ovat muuttuneet. Kun vain tarvittavat solut talletetaan, onnistuu tietokantaoperaatio nopeasti.

3.3 Ulkoasu

Tiedot on sijoitettu Excel-tauluun siten, että rivien vasemmalla reunalla on ennustettavan tuotteen nimi. Sarakkeiden yläosassa on valitut mittarit kuukausikohtaisesti valitulle aikavälille. Ennusteiden tiedot, eli mittareiden arvot, on sijoitettu rivi- ja sarakekohtaisesti tauluun.

Tietoihin porautumistoiminto on toteutettu Excelin ryhmittelytoiminnon avulla. Ryhmittelytoiminnolla voi ryhmitellä rivejä tai sarakkeita, minkä jälkeen käyttäjä voi piilottaa tai ottaa esille ne painamalla ryhmän sivussa olevaa painiketta. Ennustesovelluksessä käytettiin rivikohtaista ryhmittelyä tuotehierarkian toteuttamiseen.

Kuviossa 18 on kuvakaappaus käyttöliittymästä kehitysvaiheessa.



	A	B	C	D
1	Get data	Save to database		
2				
3	Calculation finished: 37ms	2011-4		2011-5
4		Turnover	Items sold	Turnover
5	**Test hierarchy**	10070	173	10070
6	**Test group 1**	3000	75	3000
7	**Test category 1**	1000	25	1000
8	**Test product 1**	200	5	200
9	**Test product 2**	200	4	200
10	**Test product 3**	200	4	200
11	**Test product 4**	100	6	100
12	**Test product 5**	150	3	150
13	**Test product 6**	150	3	150
14				
15	**Test category 2**	1000	25	1000
16	**Test product 7**	250	5	250
17	**Test product 8**	300	6	300
18	**Test product 9**	200	6	200
19	**Test product 10**	250	8	250
20				
21	**Test category 3**	1000	25	1000
50				
51	**Test group 2**	5520	68	5520
80	**Test group 3**	1550	30	1550
109				
110				
111				

KUVIO 18. Sovelluksen käyttöliittymä

Ulkoasuun sisältyy myös erilaisiin toimintoihin kuluvan ajan näyttäminen. Tämä toiminnallisuus on tärkeä sen takia, että asiakas näkee helpommin sovelluksesta saatavan nopeusedun. Esimerkiksi tietokantahaun tai tietojen muokkaamisen jälkeen näytetään kulunut aika.

4 YHTEENVETO

Tietovarastot ovat tärkeitä yrityksen toiminnan ohjauksessa. Tietovarastojen tekeminen vaatii kuitenkin aikaa ja resursseja, joten aivan pienille yrityksille se ei välttämättä sovellu kokonaisuudessaan.

Tietojen esittäminen tehdään usein käyttäen faktoja ja dimensioita. Tämä mahdollistaa tietoon porautumisen ja tiedon helpon etsimisen.

Tietovaraston haasteellisin osa on ETL-vaiheen toteuttaminen, eli tietojen integroiminen operatiivisista tietokannoista yhteen tietovarastoon. Myös tiedon esittäminen käyttäjälle vaatii aikaa ja asiantuntemusta.

Tietovaraston suurimmat edut teknisessä mielessä ovat tiedon helppo hakeminen, sekä hyvä suorituskyky. Käyttäjän kannalta etuna on, että kaikki historiatiedot säilyvät.

Ennustussovellus POC-projekti onnistui odotusten mukaisesti. Ohjelman toiminnallisuus saatiin kuntoon ajoissa. Ajan myötä nähdään, onko asiakas riittävän tyytyväinen tilatakseen lopullisen sovelluksen. Projektin varsinaisena tavoitteena oli osoittaa, että kyseinen ennustesovellus toimii paremmin Excel-pohjaisena, jolloin asiakas tilaa kokonaisen version. Tavoitteen täyttymistä ei voi vielä arvioida tämän opinnäytetyön kirjoittamisehetkellä, koska sovellusta ei ole vielä esitelty asiakkaalle.

Ongelmia ennustesovelluksen tekemisessä ei oikeastaan ollut. Haasteita aiheuttivat tietyt tekniset seikat, mutta mikään niistä ei hidastanut projektin valmistumista.

LÄHTEET

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2009. Tutki ja kirjoita. 15. uudistettu painos. Helsinki: Tammi.

Inmon, W. H. 2002. Building The Data Warehouse Third Edition. USA. John Wiley & Sons Inc.

Kimball, R., Ross, M. 2002. Data Warehouse Toolkit Second Edition. USA. John Wiley & Sons Inc.

Wikipedia. 2011. Relational database. [viitattu 30.3.2011] Saatavissa: http://en.wikipedia.org/wiki/Relational_database

Wikipedia. 2011. SQL. [viitattu 30.3.2011] Saatavissa: http://en.wikipedia.org/wiki/Sql

Oracle, 2002. Schema Modeling Techniques. [viitattu 6.4.2011] Saatavissa: http://download.oracle.com/docs/cd/B10500_01/server.920/a96520/schemas.htm

Microsoft. 2011. Office Solutions Development Overview. [viitattu 30.3.2011] Saatavissa: http://msdn.microsoft.com/fi-fi/library/hy7c6z9k.aspx

Microsoft. 2011. VBA and Office Solutions in Visual Studio Compared. [viitattu 30.3.2011] Saatavissa: http://msdn.microsoft.com/fi-fi/library/ss11825b.aspx

Microsoft. 2011. Architecture of Document-Level Customizations. [viitattu 30.30.2011] Saatavissa: http://msdn.microsoft.com/en-us/library/zcfbd2sk(v=VS.100).aspx

Microsoft. 2011. Configuring a Computer to Develop Office Solutions [viitattu 30.30.2011] Saatavissa: http://msdn.microsoft.com/fi-fi/library/bb398242.aspx)

Microsoft. 2011. Deploying Office Solutions [viitattu 30.30.2011] Saatavissa: http://msdn.microsoft.com/fi-fi/library/bb386179.aspx